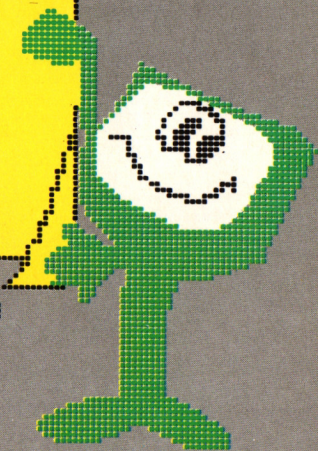
20 VIDEOLEZIONI DI BASIC
PER IMPARARE CON L'MSX

VIDEOBASIC MSX
Pubblicazione quattordicinale
edita dal Gruppo Editoriale Jackson
Direttore Responsabile:
Giampietro Zanga
Direttore e Coordinatore
Editoriale: Roberto Pancaldi
Autore: Softidea -
Via Indipendenza 88-90 - Como
Redazione software:
Michele Casertelli
Francesco Franceschini
Progetto grafico:
Studio Nuovidea - via Longhi, 16 - Milano
Impaginazione:
Moreno Confalone
Illustrazioni:
Cinzia Ferrari, Silvano Scolari
Fotografie:
Marcello Longhini
Distribuzione: SODIP
Via Zuretti, 12 - Milano
Fotocomposizione: Lineacomp S.r.l.
Via Rosellini, 12 - Milano
Stampa: Grafika '78
Via Trieste, 20 - Pioltello (MI)
Direzione e Redazione:
Via Rosellini, 12 - 20124 Milano
Tel. 02/6880951/5

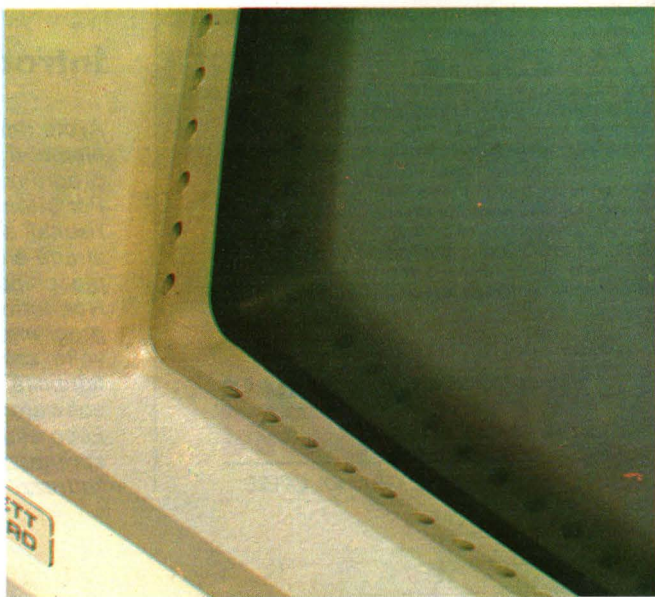
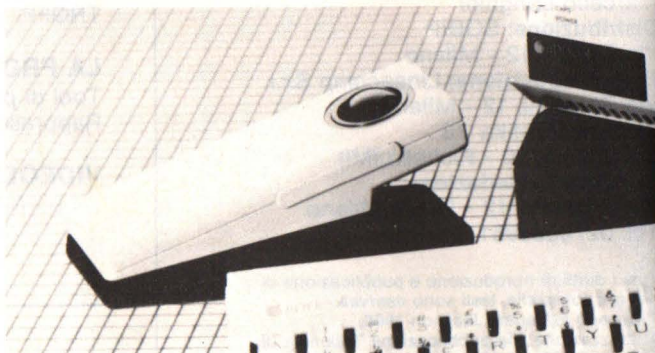
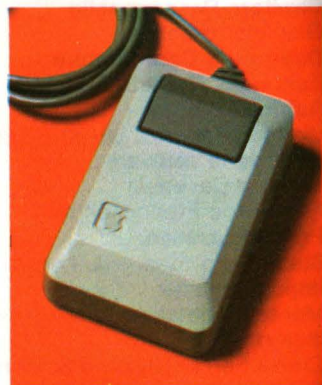
Autorizzazione alla pubblicazione Tribunale di Milano n° 422 del 22-9-1984
Spedizione in abbonamento postale Gruppo II/70 (autorizzazione della Direzione Provinciale delle PPTT di Milano).
Prezzo del fascicolo L. 8.000
Abbonamento comprensivo di 5 raccoglitori L. 165.000
I versamenti vanno indirizzati a: Gruppo Editoriale Jackson S.p.A. - Via Rosellini, 12 20124 Milano, mediante emissione di assegno bancario o cartolina vaglia oppure utilizzando il c.c.p. n° 11666203.
I numeri arretrati possono essere richiesti direttamente all'editore inviando L. 10.000 cdu. mediante assegno bancario o vaglia postale o francobolli.
Non vengono effettuate spedizioni contrassegno.

SOMMARIO

Introduzione

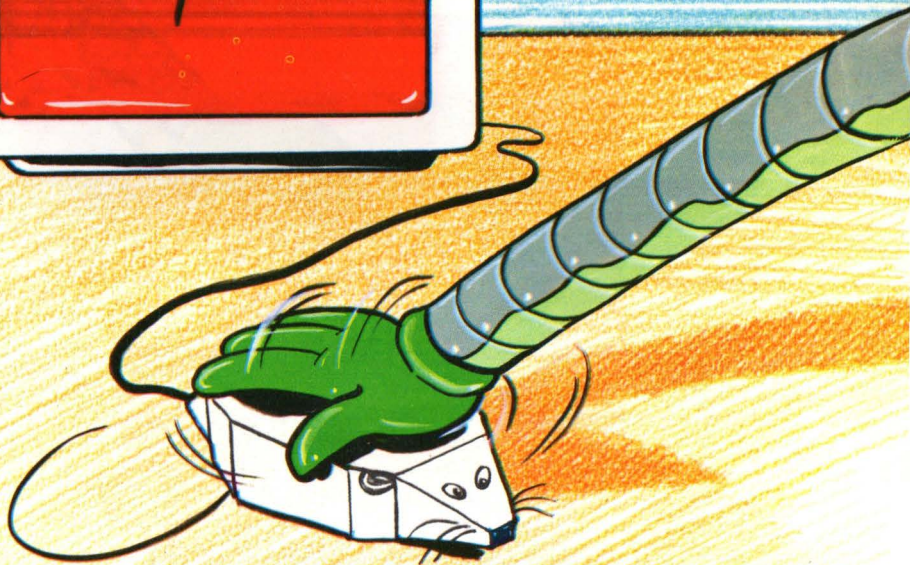
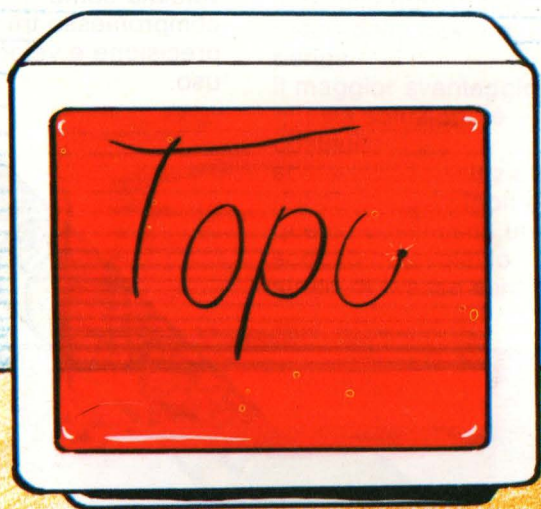
Avete ormai a disposizione tutti gli elementi per essere dei buoni programmatori in BASIC.
Per ottenere dal computer, però, risultati ancora più strabilianti la strada è una sola: il linguaggio macchina.
Non tutto per fortuna deve essere programmato autonomamente; molte volte, anzi, è preferibile in termini di fattibilità, tempi e costi, acquistare software standard.
Ecco allora word processor, spreadsheet, data base e altri "attrezzi" di grande utilità.

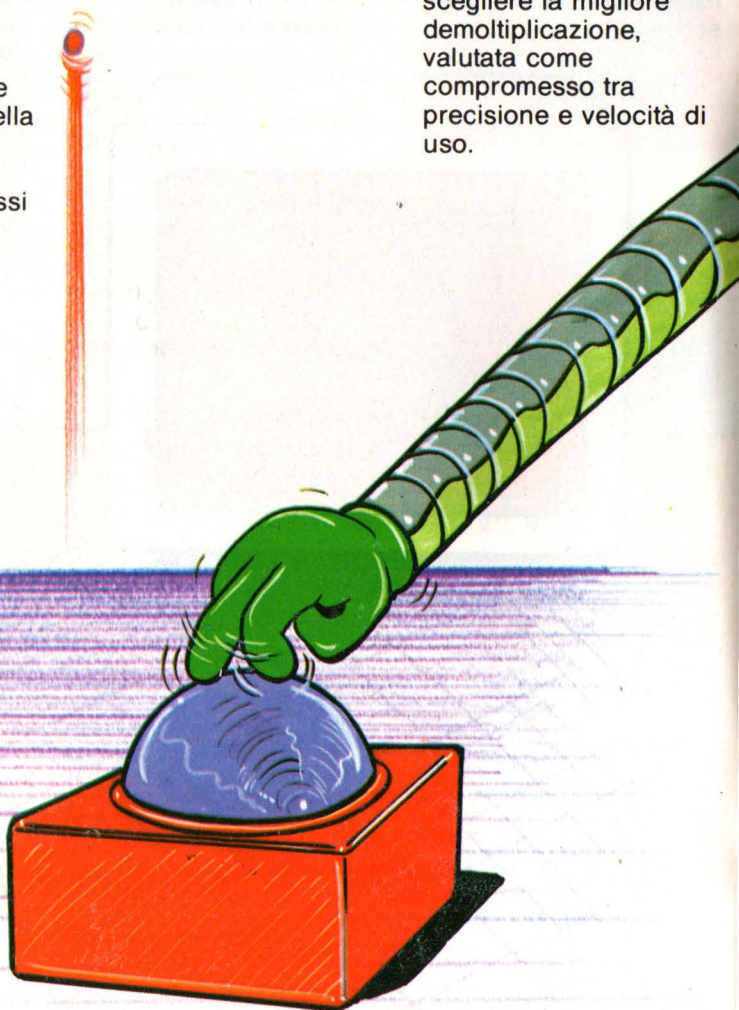
Moderni dispositivi di input

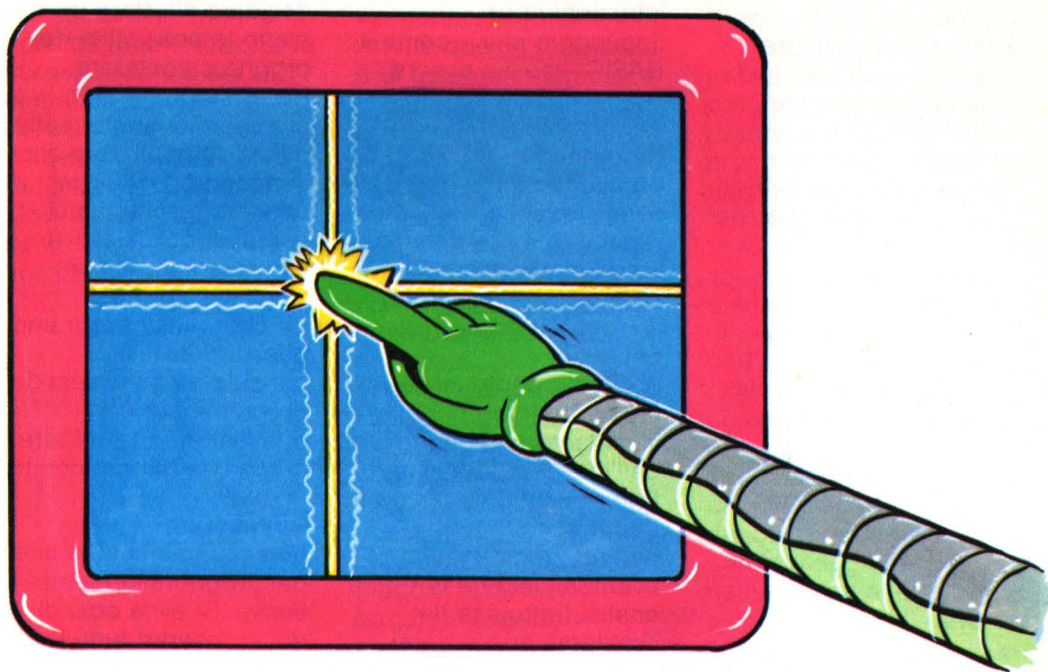
Come abbiamo già avuto modo di vedere, per qualsiasi computer la tastiera costituisce un dispositivo di input estremamente importante, anzi fondamentale.
Esistono tuttavia numerose altre periferiche di input che consentono, di fornire al computer in modo semplice e immediato, i dati che si desidera immettere nella memoria.
Da quando la grafica è entrata prepotentemente nel campo dei personal computer, si sono sviluppati nuovi dispositivi per rendere più semplice ed immediato il colloquio uomo-macchina.
I più diffusi sono il mouse, la trackball e il touch-screen.

MOUSE

Il mouse ("topolino") è della trackball capovolta. La sfera viene infatti posta in rotazione spostando l'intero dispositivo su un piano liscio (per esempio una scrivania).

Rispetto alla trackball, il mouse richiede uno spazio maggiore, essendo necessaria una certa libertà di movimento per la manovra: in compenso l'operazione risulta più intuitiva, in quanto lo spostamento del cursore sullo schermo ricalca fedelmente quello imposto con la mano. È inoltre possibile muovere il cursore tenendo contemporaneamente premuto il pulsante (o i pulsanti) del mouse, cosa impossibile con la trackball.

Per i suoi vantaggi di precisione e facilità di impiego, il mouse si sta attualmente affermando come la principale periferica di input della nuova generazione di personal computer.

TRACKBALL

La trackball (letteralmente "sfera tracciante") è una periferica costituita da una sfera liscia, completamente libera di ruotare in un supporto fisso; viene posta in rotazione utilizzando il palmo della mano. Due rullini posti a fianco della sfera (all'interno del supporto) rilevano la rotazione lungo due assi ortogonali e la convertono in una serie di impulsi elettrici, in numero proporzionale all'angolo di rotazione. Dato che - a differenza di joystick e paddle - non esiste fine corsa, è possibile ottenere tutta la precisione che si desidera, anche se talvolta questo vantaggio va a scapito della velocità di utilizzo.
È pertanto compito del software di "pilotaggio" scegliere la migliore demoltiplicazione, valutata come compromesso tra precisione e velocità di uso.

TOUCH-SCREEN

L'idea del touch-screen è molto simile a quella della penna luminosa: touch-screen significa infatti "video sensibile al tatto".
In realtà non è lo schermo video ad essere sensibile al tocco dell'operatore, ma un foglio trasparente che ricopre lo schermo stesso. Un reticolo di sensori ottici rende sensibile questo foglio al tocco delle dita o di una penna.
Il maggior svantaggio di questa soluzione è costituito dalla scarsissima risoluzione del dispositivo. Inoltre, e questo costituisce uno scoglio notevole, lo strato di grasso sempre presente sulla pelle finisce con l'accumularsi sul video, pregiudicando la lettura dello schermo.
Di contro, l'uso di questa periferica risulta estremamente naturale ed intuitivo, potendo quindi essere utilizzata anche da persone non troppo esperte.

Software di ausilio

Qualunque computer, per quanto sofisticato e perfezionato, da solo non è in grado di operare: occorre infatti abbinare alle caratteristiche hardware della macchina il software appropriato alla risoluzione del problema (o dei problemi) che si intende risolvere.
Per perseguire questo obiettivo, tuttavia, il programmatore deve compiere un insieme di passi ed operazioni variabili di volta in volta.
Occorre quindi analizzare e valutare sotto ogni minimo aspetto il problema, esaminando con attenzione tutte le difficoltà da superare.
Una volta individuato nei termini generali il problema, le varie funzioni da svolgere vengono man mano evidenziate in appositi schemi (comunemente definiti - ormai lo sai benissimo - diagrammi a blocchi), che individuano la successione logica delle azioni, con le possibili situazioni alternative e gli eventuali momenti di decisione. In questa fase si decide anche se l'intero insieme di funzioni da realizzare verrà concentrato in un unico programma oppure suddiviso in più programmi e relativi sottoprogrammi.
Una volta stabiliti i diagrammi a blocchi il programma passa infine alla vera e propria fase di programmazione, realizzata mediante la scrittura delle varie istruzioni in un linguaggio che - come il BASIC - possa essere "capito" dalla macchina.
Terminata la programmazione si provvede quindi (dopo aver preventivamente registrato il programma su un sicuro supporto magnetico a avviare la successiva fase di collaudo.
A questo punto inizia il momento forse più stimolante e impegnativo dell'intero procedimento.
È in questa fase che tutti i nodi vengono al pettine, evidenziando le eventuali lacune che una analisi frettolosa del problema può essersi lasciata alle spalle.
In tale operazione possono sovente essere di aiuto al programmatore quegli insiemi di programmi, comunemente definiti come "software di ausilio", che permettono di ispezionare, valutare e correggere gli errori in modo molto più veloce ed agevole.
I programmi di ausilio più comuni riguardano e permettono operazioni a prima vista molto banali: al momento opportuno essi diventano comunque pressoché indispensabili, specialmente quando si desidera sfruttare in pieno le possibilità del proprio calcolatore.
Ne esistono molti in commercio; anche sulle riviste specializzate essi vengono spesso presi in considerazione. I più completi TOOL-KIT (tool significa letteralmente "utensile", proprio perché questi programmi possono essere considerati gli arnesi del programmatore) consentono di svolgere numerose operazioni, tra le quali:

- **numerare** automaticamente le linee del programma mentre si scrive. Si evita così di dover inserire tutte le

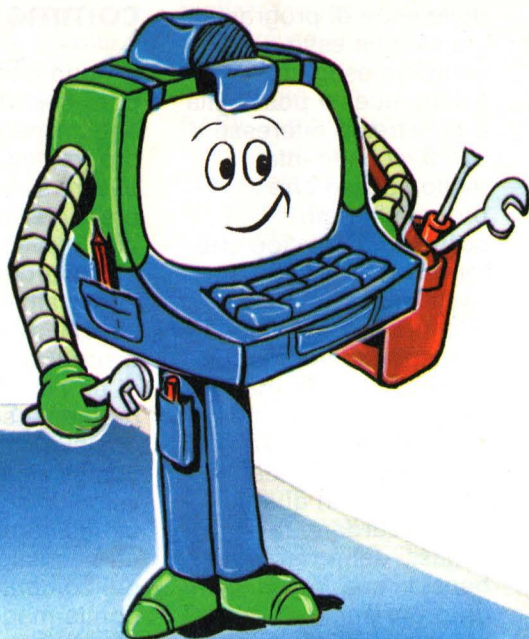
volte i numeri di linea; naturalmente, è possibile specificare il passo - cioè l'incremento - che si desidera avere tra una linea e l'altra;

- **rinumerare** un programma che abbia subìto molti aggiustamenti. Questa è una delle possibilità più apprezzate dai programmatori, che spesso - se non esistessero questi programmi - non avrebbero più modo di inserire nuove istruzioni. È chiaro che la rinumerazione delle linee deve implicare - per lo meno in un programma professionale - anche la corretta rinumerazione in corrispondenza dei vari comandi GOTO e GOSUB;
- **cancellare** blocchi di programma. Come gli MSX, molti elaboratori permettono di effettuare questa azione. Con "delete" (cancellazione) è possibile dare comandi del tipo "delete 10-130" (cancella tutte le linee comprese tra la 10 e la 130);
- **ricercare** una particolare stringa di caratteri nel programma. Si evita in questo modo di dover faticosamente ricercare la stringa (per esempio un nome di variabile) per tutta la lunghezza del listato;
- **sostituire** automaticamente una stringa di caratteri con un'altra;
- **visualizzare** i numeri delle linee di programma via via che esse vengono eseguite.

Anche questa possibilità è di estremo interesse e utilità: accade infatti molto spesso che il programma giri correttamente, cioè che fornisca dei risultati in uscita, ma che questi non siano assolutamente corrispondenti a quanto dovrebbe realmente accadere.
Sono questi gli errori più subdoli e difficili da individuare, in quanto non riguardano la sintassi del programma, bensì la sua logica. Con un programma di "trace" (traccia) è allora possibile seguire - attraverso la visualizzazione progressiva dei numeri di linea che vengono eseguiti - lo svolgimento del programma in tempo reale, cioè nel momento stesso in cui esso sta funzionando. Molte volte è sufficiente un'occhiata a questi numeri per essere in grado di risolvere un guaio provocato da un errato salto di riga o da una mancata successione di istruzioni.

Supporti commerciali

Proprio come accade per i capi di abbigliamento, i programmi possono essere su misura o preconfezionati. Questi ultimi, ovviamente, costano meno.
I programmi specificamente progettati e scritti per la soluzione di un determinato problema sono quelli che svolgono il loro compito con la massima velocità, con il migliore sfruttamento delle caratteristiche tecniche del computer utilizzato e con la maggiore rispondenza alle esigenze di chi li deve adoperare.

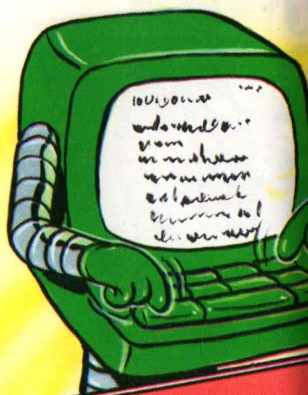
I vantaggi di questi programmi possono essere riassunti in pochi ma importanti punti:
— sono realizzati professionalmente e collaudati;
— sono pronti immediatamente (non richiedono cioè lunghi tempi di attesa);
— hanno un prezzo abbastanza contenuto.
Analizzando anche il rovescio della medaglia, lo svantaggio principale di questi programmi è che essi non possono soddisfare al cento per cento le esigenze di chi li usa. Esiste infatti sempre un certo margine di "insoddisfazione", dovuto al fatto che un programma progettato per risolvere un problema nel suo complesso non può tener conto di tutti i casi particolari.
Un programma di contabilità - per esempio - non potrà essere identico per un artigiano o per una media industria: per questo molti programmi sono allora "aperti", cioè permettono un margine più o meno ampio di modificabilità, per eventuali "personalizzazioni".
Ci sono invece esigenze che sono realmente comuni ad un numero molto vasto di utenti,

come la scrittura e l'archiviazione dei dati. Infatti i più importanti "pacchetti applicativi" disponibili attualmente sul mercato sono: i word processor, i fogli elettronici e i data base.

Word processor

Un programma per l'elaborazione dei testi (dall'inglese word processor) è una delle possibili applicazioni che, da sola, può addirittura giustificare l'acquisto di un personal computer.
La caratteristica principale di un word processor è infatti quella di trasformare l'elaboratore in una macchina da scrivere molto speciale. Permette infatti di scrivere di getto, quasi alla rinfusa, tutto ciò che si desidera, con la possibilità di modificare qualsiasi parte del testo senza dover riscrivere tutto da capo al momento di battere la bella copia e consentendo di ottenere con la stessa qualità di stampa un numero illimitato di copie. Il suo

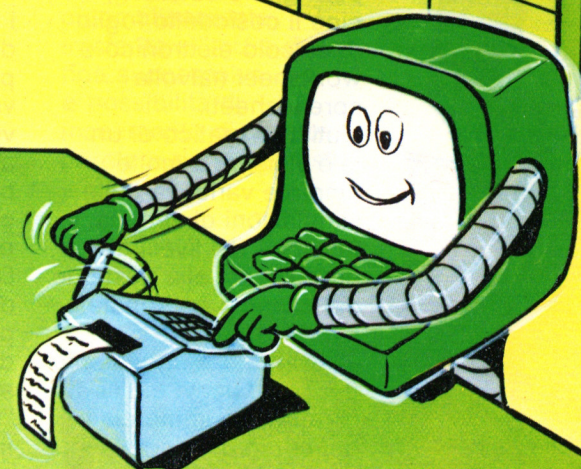
uso abolisce quindi in maniera definitiva gomme, correttori, carta carbone e cose simili. Un sistema di trattamento della parola gestisce inoltre automaticamente l'allineamento delle parole sia all'interno delle righe che dei margini, i quali possono essere variati a piacimento con pochi ed elementari comandi. Ma la caratteristica fondamentale di un programma word processor è comunque quella di permettere la correzione dinamica di tutto ciò che si è scritto: si può cioè "tornare indietro" e correggere direttamente sul video, lasciando al computer e alla sua stampante il compito di riscrivere l'intero testo su carta.

Le funzioni più importanti e necessarie di un sistema per il trattamento di testi sono:

- a capo automatico. Non è quindi necessario controllare il fine riga, poiché le parole eccedenti la capienza della stessa vengono automaticamente trasferite all'inizio della riga successsiva;
- possibilità di inserimento di caratteri, parole o righe nel testo già esistente, per correzioni o aggiunte;
- cancellazione di caratteri, parole e frasi, con eliminazione automatica dello spazio resosi disponibile nella riga;
- possibilità di reimpaginare automaticamente il testo, sia perché si è voluto variare il numero di caratteri per riga sia perché si è variata la spaziatura tra le righe;

● possibilità della ricerca automatica di una parola o di un gruppo di parole nell'intero testo;
● possibilità di copiare e trasferire parti di testo in altra parte dello stesso documento.
In programmi maggiormente evoluti esistono altre funzioni, più particolari di quelle appena accennate, ma non per questo di minore utilità: il principio generale di funzionamento resta comunque sempre lo stesso. Naturalmente, qualsiasi word processor permette di salvare il testo su un supporto magnetico.

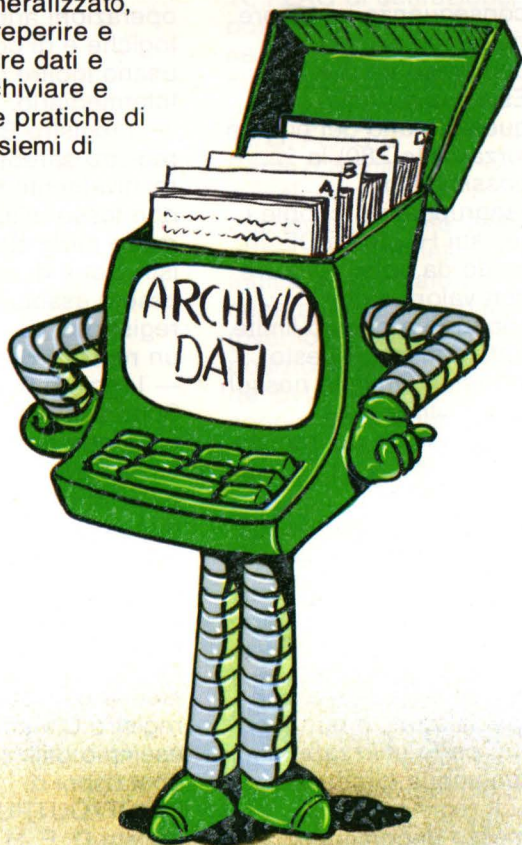
Fogli elettronici

Una delle applicazioni che ha avuto maggior fortuna e diffusione nel settore degli home e personal computer è sicuramente il trattamento automatico dei prospetti di tipo tabellare (quelli, per intenderci, organizzati per righe e colonne), cioè il cosiddetto foglio di calcolo elettronico o worksheet (talvolta spreadsheet).
L'utilizzo pratico di un tale sistema è molto ampio e vario: va dalle simulazioni di vendita, in funzione dei livelli di produzione, alla gestione del bilancio personale o familiare, oppure all'esame dell'andamento dei costi, ricavi e utili di un'attività commerciale nel corso dei dodici mesi dell'anno.
Ma vediamo brevemente come funziona.
Si tratta di un insieme di righe e colonne individuate da un codice (ad esempio, lettere per le colonne e numeri per le righe, proprio come nel gioco della battaglia navale). Ciascuna coordinata può contenere un dato numerico, una formula che collega ed elabora algebricamente o logicamente più dati, o un insieme di lettere qualsiasi, quali quelle componenti un titolo. Tra le diverse informazioni che possono comparire in due o più caselle possono essere definite delle relazioni analitiche, algebriche o di altro tipo (per esempio statistico).
Ciò fatto, è sufficiente introdurre tramite tastiera i vari valori nelle caselle definite come variabili principali, perché tutte le variabili dipendenti vengano automaticamente calcolate dal programma e fatte comparire nelle caselle pertinenti.
Da questa caratteristica deriva la capacità dei pacchetti di spreadsheet di fungere da potenti strumenti di "simulazione" per modelli di tipo statistico o finanziario. L'acquisto di un foglio di lavoro elettronico è pertanto consigliabile in tutti i casi in cui esista una esigenza,

GENN.
FEBB.
MARZO
APR.
MAGGIO
GIU.
AFFITTO
CASA
GAS
BENZINA
LUCE
TOTALE

ragionevolmente frequente, di elaborare prospetti tabellari con dati eventualmente tra loro correlati.

Data base

I data base (o, con una libera traduzione italiana, i programmi per l'organizzazione di dati) sono programmi di utilizzo generalizzato, adatti per reperire e memorizzare dati e notizie, archiviare e classificare pratiche di ufficio o insiemi di informazioni e così via. Un data base permette di gestire elettronicamente qualsiasi tipo di archivio di informazioni, disposto e strutturato secondo il desiderio di chi utilizza il programma. Un esempio renderà l'idea nel migliore dei modi. Supponiamo di voler comporre una rubrica telefonica elettronica, contenente - oltre al nome e al cognome (ed ovviamente al numero telefonico) delle varie persone - anche l'indirizzo e la città di residenza.

Con il data base dopo aver introdotto gli elementi è possibile ordinare la lista delle persone, per esempio in ordine alfabetico oppure per zone di residenza, per prefisso telefonico o per qualsiasi altra "chiave".
È inoltre immediatamente ricercabile, in modo assolutamente semplice e automatico, "la persona che ha prefisso X e abitazione con numero civico Y".
Per tornare a un discorso generale, le applicazioni di un data base sono praticamente infinite: tante quanti possono essere gli utilizzi di archivi di informazione. La cosa importante di questi programmi è che essi permettono la gestione di qualsiasi tipo di informazione, non importa quanto strana o complicata, ricercandola ed aggiornandola con la massima rapidità e automazione.

Linguaggio macchina: i registri

Il microprocessore montato sul tuo MSX (ma, più in generale, tutte le CPU) dispone di un certo numero di speciali "locazioni", chiamate registri, che per il programmatore in linguaggio macchina possono essere paragonate alle variabili del programmatore in BASIC.

Sullo Z80 esistono circa 10 registri, ciascuno dei quali viene indicato mediante una lettera dell'alfabeto; tra essi i principali sono:

A, B, C, D, E, H, L

Ogni registro può essere considerato come una particolare locazione della memoria, contenente di conseguenza un valore numerico sempre compreso tra 0 e 255. Esiste comunque (e questo è uno dei punti di forza dello Z80) la possibilità di raggruppare a coppie i registri HL, BC e DE, in modo da poter lavorare con valori anche superiori a questo limite. Naturalmente, questo fatto non limita in nessun caso l'utilizzo indipendente dei registri. Passeremo ora in esame - con maggior attenzione - alcuni tra i diversi registri dello Z80, parlando un po' delle funzioni che li coinvolgono, perché alcuni di loro, e lo vedremo tra breve, sono specializzati, e non li si può certo utilizzare in qualunque circostanza.

— Il registro A. Più spesso denominato come "accumulatore", è il registro più importante del microprocessore. È lui che "accumula" la maggior parte dei compiti nel funzionamento della CPU. In compenso può accettare - a differenza degli altri registri - tutti i modi di indirizzamento disponibili (i modi di indirizzamento ti verranno presentati tra breve). Tutte le operazioni aritmetiche, logiche e di confronto lo usano inoltre come intermediario.

— Il registro B. Questo registro sarebbe relativamente banale, se non fosse utilizzato in modo particolare in una istruzione di salto. Può essere associato con il registro C per formare un registro doppio.

— I registri C, D, E, H, L. Sono semplici registri a 8 bit, che possono essere uniti a coppie: BC, DE, HL. Ad essi si può accedere molto velocemente e vengono in generale utilizzati per conservare i dati. Ognuno di essi è privilegiato riguardo ad alcune operazioni. Il registro C viene per esempio utilizzato nelle operazioni di INPUT/OUTPUT. I registri D, E, H, L vengono invece utilizzati

per il trasferimento di dati.
— Il registro F. Questo registro è il "portabandiera" del microprocessore, nel senso che ciascuno dei suoi bit indica il tipo di risultato conseguente a una certa operazione. I bit del registro di stato (numerati da 0 a 7, 7 quello più a sinistra e 0 quello più a destra) sono specificati con nomi particolari che passiamo brevemente ad esaminare:
.C, FLAG di carry, posizione 0, contiene il bit più significativo dell'ultima operazione eseguita. Il bit di carry assume cioè il riporto generato, nel corso di un'addizione, dalla somma dei due bit più significativi.
.N, FLAG di sottrazione, posizione 1, contiene 1, se l'ultima operazione è stata una sottrazione o un decremento.
.P/O, FLAG di Parità/Overflow, posizione 2, ha un duplice significato. Nelle operazioni aritmetiche indica se c'è stato un superamento di capacità (overflow), invadendo la posizione più a sinistra del byte, usata per il segno. Nelle operazioni logiche indica invece la parità (diventa 1 quando si verifica parità pari).
.AC, FLAG di Carry Ausiliario, posizione 4, viene utilizzato nelle operazioni aritmetiche eseguite su numeri espressi secondo una particolare codifica. È un "mezzo-carry", da cui il suo nome. In effetti dà le stesse indicazioni del bit di carry, ma su un valore di soli 4 bit.
.Z, FLAG di zero, posizione 6, si trova a 1, se l'ultima operazione ha dato risultato zero.
.S, FLAG di Segno, posizione 7, ha il valore del bit più significativo del risultato dell'ultima operazione logica o aritmetica eseguita dal microprocessore.

Tecniche di indirizzamento

Lo Z80 non dispone di una grossa varietà di istruzioni (complessivamente sono 67). Sotto questo aspetto numerose altre CPU risultano nettamente superiori, arrivando facilmente ad avere un numero di istruzioni eseguibili quasi doppio. Ciò nonostante, lo Z80 è uno dei microprocessori più diffusi.
La ragione di questo fatto è molto semplice: tutto è dovuto all'ampia possibilità di tecniche di indirizzamento disponibili che incrementa il numero effettivo delle istruzioni ad oltre 600.
Vediamo innanzitutto cosa intendiamo con l'espressione "tecniche di indirizzamento".
Abbiamo già detto che l'Assembler è un

linguaggio a basso livello, scarsamente (anzi, per nulla) strutturato e dotato di forme logiche veramente minime. In altre parole, l'Assembler si avvale di istruzioni con estrema semplicità operativa ed esecutiva: tutto viene difatti basato sui numeri e sulle locazioni di memoria.
Proprio in virtù di questo fatto, per poter lavorare al meglio in qualsiasi parte della memoria, i costruttori di microprocessori cercano di introdurre la massima flessibilità nelle operazioni eseguibili, fornendo alla CPU le più vaste tecniche di indirizzamento.
L'indirizzamento fa pertanto riferimento al modo in cui deve essere considerato l'operando in una certa istruzione, influenzando di conseguenza anche il risultato della operazione stessa. Esamineremo adesso, uno per uno, i diversi indirizzamenti disponibili, valutandone le possibilità, gli utilizzi, i vantaggi e gli svantaggi.

Indirizzamento diretto su registro

L'operando è un registro. Così

```
LD A, B
```

significa "carica in A il contenuto di B". Le istruzioni di indirizzamento diretto su registro occupano solamente un byte: esse quindi non solo sono brevi, ma consentono pure un'alta velocità. Il tempo necessario per la loro esecuzione è infatti limitato a soli 4 cicli di clock, che nel tuo MSX corrispondono a meno di un milionesimo di secondo. Nota: per ciclo di clock si intende il tempo necessario alla CPU per svolgere un'operazione elementare. Tale tempo viene scandito inflessibilmente da un oscillatore (una specie di orologio al quarzo in miniatura). Tieni presente che - a titolo di informazione - lo Z80 esegue più di 4 milioni di operazioni elementari in un secondo: e questo per tutto il tempo che tieni acceso il computer!

Indirizzamento implicito

Il codice operativo "implica" l'indirizzo dell'operando o di uno degli operandi. Per esempio: RRA serve per eseguire una operazione di rotazione verso destra di un bit del registro A. BIT 0, A esamina invece il bit di posizione 0 del registro A.

Indirizzamento immediato

Questa forma di indirizzamento è utilizzata per caricare con uno specifico valore l'accumulatore o qualche altro registro. Tutte le istruzioni nel modo a indirizzamento immediato sono perciò lunghe due byte. Il primo contiene il codice operativo e il secondo contiene la costante numerica da caricare nel registro o da utilizzare per una istruzione aritmetica o logica. Per esempio, se noi volessimo caricare

nell'accumulatore il valore 160 (esadecimale A0) in modo immediato, potremmo scrivere:

```
LD A, A0H
```

Questa istruzione dice: "carica nell'accumulatore il valore esadecimale A0".

Indirizzamento diretto

L'indirizzamento diretto è il modo in cui i dati sono normalmente recuperati dalla memoria. È specificato dal codice operativo, seguito da un indirizzo a 16 bit racchiuso tra parentesi. L'indirizzamento diretto richiede pertanto tre byte. Un esempio di indirizzamento diretto è:

```
LD A, (1234H)
```

Questa istruzione specifica che il contenuto della locazione di memoria numero 1234 deve essere memorizzato nell'accumulatore.
Lo svantaggio dell'indirizzamento diretto è di richiedere un'istruzione di tre byte: esso è quindi un modo abbastanza lento.

Indirizzamento indiretto tramite registri

Per migliorare l'efficienza dell'indirizzamento diretto viene allora reso disponibile un altro modo, quello indiretto tramite registri.
Con questo indirizzamento una coppia di registri contiene l'indirizzo nel quale si trova il valore da elaborare, cioè l'operando; la coppia di registri viene indicata tra parentesi. In gergo informatico si dice che i registri "puntano" la locazione di memoria.
Un esempio di questo modo è:

```
LD A, (HL)
```

che significa "carica in A il contenuto del byte il cui indirizzo si trova in HL".
L'indirizzamento indiretto tramite registri è nettamente più veloce di quello indiretto, in quanto la CPU non deve leggere l'indirizzo di memoria, visto che esso si trova già nei registri.
Naturalmente, occorre caricare nella coppia di registri l'indirizzo desiderato, e quindi questo metodo diventa vantaggioso solo se l'indirizzo viene utilizzato più volte.

Indirizzamento relativo

Questo metodo di indirizzamento è spesso utilizzato per le istruzioni di salto (cioè per i comandi che trasferiscono l'esecuzione da un punto all'altro del programma). Per definizione l'indirizzamento relativo utilizza due byte: il primo è un'istruzione di salto, mentre il secondo specifica lo spostamento e il suo segno. Cosa significa? Supponiamo per esempio di voler mandare l'esecuzione da un certo punto del programma a un altro. Per fare questo dobbiamo specificare (oltre naturalmente all'istruzione di salto) anche il numero di locazioni di cui vogliamo spostarci. Poiché lo spostamento può essere sia positivo che negativo (cioè può avvenire sia in avanti che indietro), l'istruzione di salto - che al massimo deve interessare 255 locazioni - può essere al limite negativa di 128 (salto all'indietro di 128 locazioni), oppure positiva di 127 (salto in avanti di 127 locazioni). Naturalmente, il più grosso svantaggio di questo modo è che non permette salti superiori alle 128 locazioni. Tuttavia, poiché la maggior parte dei cicli tende a essere breve, la diramazione può essere utilizzata quasi sempre e aiuta a risolvere il problema.

Indirizzamento indicizzato

L'indirizzamento indicizzato è una tecnica particolarmente pratica per accedere, uno dopo l'altro, ai dati contenuti in un gruppo di locazioni. Il principio è che l'indirizzo desiderato viene calcolato sommando all'indirizzo contenuto in uno dei registri indice (i registri indice IX e IY sono altre due coppie di registri dello Z80) il valore dell'operando. Così

```
LD E, (IX+5)
```

carica nel registro E il contenuto del byte avente l'indirizzo formato sommando il numero 5 al contenuto della coppia di registri IX. Tutti i modi di indirizzamento (per semplicità ne abbiamo tralasciati alcuni) richiedono comunque, per essere veramente capiti ed utilizzati al meglio, parecchio tempo e (soprattutto) molto lavoro di programmazione. Gli esempi che vedremo tra breve illustreranno qualche possibile applicazione di queste tecniche.

Incremento/ decremento

La più semplice espressione aritmetica che lo Z80 è in grado di eseguire è l'operazione algebrica di addizione e sottrazione del valore 1 dal valore contenuto in un registro. A prima vista questa possibilità non sembra aprire orizzonti sconfinati: in realtà la disponibilità di una simile operazione permette la costruzione di strutture ben più complesse, come per esempio i cicli FOR...NEXT del BASIC.
L'importanza dei comandi di addizione e sottrazione è talmente grande che esistono addirittura due istruzioni specifiche per portare a termine questi compiti. Di ciascuna di queste esistono due diverse forme, disponibili con differenti modi di indirizzamento:

ADD istruzione di somma senza riporto
ADC istruzione di somma con riporto
SUB istruzione di sottrazione senza riporto
SBC istruzione di sottrazione con riporto

Tutte queste istruzioni possono influenzare alcuni bit del registro di stato, precisamente il bit di segno negativo e il bit zero.
Il bit di segno viene settato, se il bit più significativo del registro (cioè quello della posizione 7) viene posto a 1 a seguito di incremento o decremento, altrimenti vale zero.
Il FLAG di zero viene invece settato soltanto se, a seguito di una operazione, il registro chiamato in causa contiene il valore zero.
Per esempio, incrementando di 1 il valore FFH (255 esadecimale) contenuto in un certo registro, si porterà a 00H il valore del registro, verrà posto a 1 il FLAG di zero (Z=1) e si annullerà il FLAG di segno negativo. D'altro canto, decrementando un registro contenente 00H, si porterà il valore del registro a FFH, ponendo contemporaneamente a 1 il FLAG di segno e a zero il FLAG di zero.

I cicli

I cicli sono una parte importantissima - anzi, vitale - in qualsiasi linguaggio di programmazione: in Assembler lo sono ancor di più, poiché molte istruzioni, che in linguaggio ad alto livello richiedono un unico comando, per la loro esecuzione, in codice macchina necessitano invece di sequenze e ripetizioni più o meno lunghe.

Prima di passare alla dimostrazione pratica dell'uso dei cicli in Assembler occorre però introdurre altri due gruppi di istruzioni: quelle di confronto e quelle di salto.

I confronti

È possibile confrontare il valore contenuto nell'accumulatore dello Z80 con il contenuto di una locazione della memoria (oppure con un valore numerico qualsiasi) mediante le istruzioni

CP confronta il valore
CPI confronta il valore con incremento
CPD confronta il valore con decremento

Il risultato del confronto altera i FLAG di zero (Z), di segno (N) e di riporto, o carry, (C) del registro di stato. In che modo? La risposta è in questa breve tabella:

ACCUMULATORE	CARRY	ZERO	SEGNO
minore del dato	1	0	1
uguale al dato	0	1	1
maggiore del dato	0	0	1

Confrontando per esempio il valore dell'accumulatore (supponiamo che sia 5AH) con il numero B3H, dalla tabella ricaviamo che il valore dei vari FLAG sarà:

C=1 Z=0 N=1

Sulla base di questi risultati potremo quindi prendere le decisioni più appropriate.

I salti

Una volta effettuata l'operazione di confronto può essere necessario eseguire anche un'operazione di salto (così come in BASIC scriviamo per esempio IF A>B THEN GO TO 300).
Possiamo distinguere tre gruppi di salti:
- i salti propriamente detti
- le chiamate ai sottoprogrammi
- i ritorni dai sottoprogrammi.

MA anche due modi di salto:
- assoluti
- relativi.

Un'istruzione di questo tipo:

```
JP ADR
```

si chiama salto incondizionato: essa provoca infatti il salto sistematico all'indirizzo ADR (indicando con ADR un generico indirizzo della memoria). È l'equivalente in linguaggio macchina del GO TO che si usa in BASIC. Queste altre istruzioni sono invece esempi di salti condizionati:

```
JP C,ADR      JP NC,ADR
JP Z,ADR      JP NZ,ADR
JP M,ADR      JP P,ADR
JP PE,ADR     JP PO,ADR
```

"Salto condizionato" significa che il salto deve essere effettuato solo se la condizione richiesta è soddisfatta. Naturalmente, le condizioni di salto condizionato si possono presentare in un programma soltanto dopo istruzioni che posizionano in qualche modo i FLAG. La condizione di salto viene infatti controllata proprio in base ai valori assunti da qualcuno di questi FLAG.

```
....
CP 8          ; confronta A con 8
JP Z, ADR1    ; salta se A=8
JP C, ADR2    ; salta se A<8
              ; allora A>8
.....
```

In questo esempio l'istruzione CP posiziona i flag Z e C. Il salto condizionato JPZ

controlla la presenza del flag Z e provoca il salto all'indirizzo ADR1, se Z è a 1, segnalando in questo modo che il contenuto dell'accumulatore è uguale a 8. In caso contrario il programma prosegue ed incontra l'istruzione JPC, che controlla lo stato del flag C. Se questo è a 1, si effettua un salto all'indirizzo ADR2, altrimenti possiamo essere sicuri che il contenuto di A è maggiore di 8, poiché non soddisfa i test precedenti, e il programma prosegue in sequenza.
Finora abbiamo visto come operando dell'istruzione JP un indirizzo di salto. È però possibile effettuare un salto non più soltanto a un indirizzo dato, ma anche ad un indirizzo calcolato, contenuto in uno dei registri HL, IX o IY:

```
JP (HL)
JP (IX)
JP (IY)
```

Questa modalità di salto torna molto utile in numerose occasioni.
I comandi di SALTO RELATIVO si distinguono dagli altri per una caratteristica speciale: specificano uno spostamento relativo, cioè in avanti o indietro di +127 locazioni o di -128 byte, rispetto alla posizione occupata in quel momento. Le istruzioni di salto relativo sono:

```
JR      VAL
JR C,   VAL      JR NC, VAL
JR Z,   VAL      JR NZ, VAL
```

VAL rappresenta un valore numerico che indica - in modo relativo - l'indirizzo a cui saltare.
Per esempio:

```
JR Z, - 14
```

significa: "se il risultato di confronto (che sarà già stato operato in precedenza) ha posto il flag Z a 1, allora esegui un salto all'indietro di 14 byte".
Oltre alle istruzioni di salto vero e proprio, esiste un ultimo comando di salto: CALL.
La CALL dell'Assembler è equivalente al GOSUB del BASIC.
Questa istruzione effettua cioè un salto al sottoprogramma il cui indirizzo è specificato nell'operando, non senza aver prima effettuato un salvataggio nell'area di stack dell'indirizzo di ritorno. È quest'ultimo punto che differenzia la CALL da una semplice JP, con la quale le analogie sono peraltro notevolissime.
Per poter tornare dai sottoprogrammi in linguaggio macchina, così come in BASIC è necessario ricorrere a RETURN, bisogna invece impartire un'istruzione RET. In questo caso il microprocessore estrae

un valore dall'area di stack e riporta l'esecuzione del programma all'indirizzo corrispondente al valore prelevato.
Terminata finalmente la parte teorica passiamo adesso a quella pratica, cioè all'applicazione dei concetti visti finora.

Comandi per programmare

Il Basic MSX mette a disposizione del programmatore una serie di comandi che consentono di rendere più comoda e veloce la stesura e la verifica dei programmi.
Naturalmente tali comandi non sono adatti ad essere inseriti in un programma; quando il computer li incontra arresta l'esecuzione.
Una loro conoscenza, però, può essere molto utile a chiunque voglia programmare in Basic.

DELETE

Il modo più semplice per cancellare una linea di un programma consiste nello scriverne il numero e poi premere RETURN.
Quando le linee di programma sono molte, però, questo procedimento richiede molto tempo.
Per fare più in fretta, basta scrivere delete seguito dal numero della prima e dell'ultima linea da cancellare separati da un trattino, come si fa con LIST.

```
DELETE 10-100
```

Questo comando serve a cancellare le linee comprese tra la 10 e la 100.
Se una di queste linee non esiste, verrà segnalato un errore.

```
DELETE - 100
```

Questa forma abbreviata può essere utilizzata quando la prima linea da cancellare corrisponde con la prima linea del programma.

Sintassi del progrmma

DELETE [X] [— Y]

RENUM

Molto spesso, programmando in Basic, capita di voler cambiare il numero di alcune linee: per fare spazio a nuove istruzioni, per effettuare il merge di altre linee o semplicemente per fare un po' di ordine.
Questo si può fare semplicemente con il comando RENUM.
Il comando RENUM, oltre a cambiare i numeri di linea, provvede ad aggiornare anche tutti i riferimenti ad essi, contenuti nelle istruzioni GOTO, GOSUB, RESTORE ecc., cosa che, se dovesse essere fatta a mano, sarebbe fonte di errori sicuri.
Il comando RENUM, se usato senza parametri, provvede semplicemente a rinumerare tutte le linee del programma con numeri da 10 in poi a passi di 10.
Se si vuole iniziare con un altro numero, bisogna specificarlo subito dopo la parola RENUM.

È possibile rinumerare solo una parte del programma specificando come secondo parametro il numero della prima linea a cui va cambiato il numero.
Il terzo parametro specifica l'incremento.

```
RENUM 50
```

Rinumera tutto il programma, dando alla prima linea il numero 50, con incremento di 10 per le successive.

```
RENUM 1000, 120
```

Rinumera le linee del programma a partire dalla 120 con numeri da 1000 in avanti a passi di 10.

```
RENUM 1000, 100, 20
```

Rinumera le linee a partire dalla 100 con numeri da 1000 in avanti a passi di 20.

```
RENUM 1000,, 50
```

Rinumera tutto il programma con numeri da 1000 in avanti a passi di 50.

Sintassi del comando

RENUM [[X]] [,Y] [,Z]]

AUTO

Una delle caratteristiche del Basic più noiose per il programmatore è sicuramente la necessità di numerare tutte le linee.
Il Basic MSX consente di numerare automaticamente le linee di programma man mano che vengono immesse grazie al comando AUTO.
Con questo comando, dopo l'immissione di ogni linea, viene scritto automaticamente il numero della linea successiva, con un incremento a scelta.
Se si scrive semplicemente AUTO, il computer comincerà a scrivere il numero di linea 10 e poi proseguirà a passi di 10.
Volendo, si possono specificare una linea di inizio o un incremento diversi, semplicemente scrivendo dopo la parola AUTO prima il numero di linea e poi l'incremento.

```
AUTO 100, 20
```

Genera i numeri di riga 100, 120, 140, 160 ecc.
Se la linea che deve essere immessa è già occupata dopo il numero si troverà un asterisco (*), per evitare sovrapposizioni.

Sintassi del comando

AUTO [[X] [,[Y]]]

TRON E TROFF

I comandi che abbiamo visto servono a facilitare il compito di inserire e correggere le linee del programma. Questa attività prende il nome di "editing".
Gran parte del lavoro del programmatore, però, è utilizzata per scovare gli errori; questa fase viene detta "Debugging".
TRON e TROFF sono le istruzioni che il Basic MSX mette a disposizione del programmatore per il debugging dei programmi.
TRON serve a abilitare il "TRACING" del programma.
Quando il TRACING è abilitato, il computer, ogni volta che esegue una linea di programma, ne stampa il numero tra parentesi quadre sullo schermo.
In questò modo è possibile verificare che l'ordine in cui le istruzioni vengono eseguite sia esattamente quello atteso, mettendo immediatamente in luce eventuali errori.
TROFF serve a disabilitare il tracing, che altrimenti resterà attivo fino allo spegnimento del computer.
Una limitazione nell'uso

del tracing come metodo di debugging consiste nella quantità di output prodotta, che oltretutto interferisce anche con l'output del programma. Per evitare una valanga di dati di difficile interpretazione, conviene inserire TRON e TROFF nei punti nevralgici del programma, utilizzando anche qualche STOP per ispezionare le variabili.

SCROLL A DESTRA DI UNA RIGA IN SCREEN

Si tratta di spostare (scroll) a destra di una posizione tutti i caratteri presenti su una determinata riga dello schermo.
Poiché le cose cambiano a seconda del modo di schermo è necessario fare una scelta: per questa applicazione scegliamo il modo SCREEN1.
Consideriamo ora gli elementi necessari al programma in LM.
1) Il numero della riga su cui operare.
2) La lunghezza della riga in caratteri.
3) L'indirizzo di inizio della memoria video contenente i caratteri da visualizzare.
Inoltre, poiché come già sai in linguaggio macchina nulla è scontato, occorre disporre di una routine in grado di leggere nella videoram e di un'altra capace di scrivervi. Il nostro programma dovrà infatti ricopiare ciò che si trova sulla riga, riscrivendolo più a destra.
Il Sistema Operativo in ROM,fortunatamente, dispone già di questi sottoprogrammi. Così non ci resta che servircene mediante chiamate agli indirizzi 74 (4AH) e 77 (4DH), due punti di ingresso del BIOS associati al processo video.

IL PROGRAMMA

Per consentire di scegliere dall'esterno (BASIC) quale riga "scrollare", adoperiamo una locazione di memoria per immagazzinarne il numero; la 50000.
In altre parole il contenuto della cella 50000 fornirà al programma in linguaggio macchina il numero della riga da elaborare.
La prima istruzione, infatti, tramite un INDIRIZZAMENTO DIRETTO carica tale numero nell'accumulatore.

```
LD ,(50000)
```

La seconda, poi, pone in BC il numero dei caratteri di una riga e la terza in HL l'indirizzo di inizio della memoria video.

```
LD BC,32
LD HL,6175
```

Ora il programma dispone di tutti gli elementi necessari ad individuare i caratteri appartenenti alla riga da trattare.
Nelle successive istruzioni, infatti, si somma tante volte 32 all'indirizzo iniziale quante indicate dal numero di riga.
In pratica si somma 32 ad HL e si decrementa il numero di riga fino a quando non diventa 0.

```
LOOP1 DEC A
      JP  Z,SAVEU
      ADD HL,BC
      JP  LOOP1
```

PROGRAMMAZIONE

Ora, chiamando in ROM la subroutine di lettura della videoram, si legge l'ultimo carattere della riga e lo si salva nello stack per non perderlo (SAVEUltimo); i caratteri che restano sono 32—1 (1F) e BC viene aggiornato.

```
SAVEU  CALL74
       PUSH AF
       LD   BC,31
```

Poi ha iniziato il ciclo (LOOP2) che legge, a partire dalla trentunesima posizione, il carattere e lo scrive nella successiva, decrementando poi il numero di posizione e ripetendo fino a quando quest'ultima non si è azzerata.

```
LOOP2  DEC  HL
       CALL74
       INC  HL
       CALL77
       DEC  HL
       DEC  BC
       LD   A,C
       OR   B
       JP   NZ,LOOP2
```

Questa tecnica consente di non perdere i caratteri che escono da destra: sono infatti reimmessi nella prima posizione dando luogo alla scritta rotante (wrap araund).

```
POP AF
CALL77
RET
```

Ovviamente, una chiamata al nostro programma darà origine allo spostamento verso destra di tutti i 32 caratteri di una sola posizione.
Per far ruotare completamente la riga saranno perciò necessarie 32 chiamate.
Alla fine non resta che richiamare dallo stack il carattere che prima occupava l'ultima posizione, scriverlo nella prima e tornare al programma chiamante.

INDIRIZZO	CODICE HEX	ASSEMBLY
9C40	3A 50 C3	LD A,(C350)
9C43	01 20 00	LD BC, 0020
9C46	21 1F 18	LD HL, 181F
9C49	3C	INC A
9C4A LOOP1	3D	DEC A
9C4B	CA 52 9C	JP, Z, 9C52 SAVEU
9C4E	09	ADD HL, BC
9C4F	C3 4A 9C	JP 9C4A LOOP1
9C52 SAVEU	CD 4A 00	CALL 004A
9C55	F5	PUSH AF
9C56	01 1F 00	LD BC, 001F
9C59 LOOP2	2B	DEC HL
9C5A	CD 4A 00	CALL 004A
9C5D	23	INC HL
9C5E	CD 4D 00	CALL 004D
9C61	2B	DEC HL
9C62	0B	DEC BC
9C63	79	LD A,C
9C64	B0	OR B
9C65	C2 59 9C	JP NZ, 9C59 LOOPZ
9C68	F1	POP AF
9C69	CD 4D 00	CALL 004D
9C6C	C9	RET

Caricatore BASIC per codici esadecimali

Molto spesso, quando si propone un programma in linguaggio macchina, si ricorre alla pubblicazione del suo disassemblato.
In tutti questi casi, oltre al mnemonico, appaiono gli indirizzi ed i codici del programma in esadecimale.
Per evitare di effettuare di volta in volta la conversione in decimale, è sufficiente predisporre un caricatore che nelle linee DATA accetti dei valori esadecimali e li gestisca automaticamente.

Questo è esattamente lo scopo del programma.
Supponiamo di memorizzare i codici del linguaggio macchina a partire dall'indirizzo 40000.
Per riservare spazio sicuro è necessario limitare l'area del BASIC fino alla locazione 39999.
Con la funzione VAL utilizzata in modo originale provvediamo alla conversione e, per riconoscere l'ultimo dei codici di introdurre, ricorriamo ad un numero esadecimale di tre cifre, sicuramente maggiore di 255.

```
10 CLEAR200, 39999! : RESTORE : IN=40000!
20 READ A$
30 N=VAL("&H" + A$) : IFN>255THEN END ELSE
    POKEIN,N
40 IN=IN + 1: GOTO20
100 DATA (IN ESADECIMALE) .....
```

WRAP AROUND

Ecco un programma per utilizzare la routine in linguaggio macchina di scroll a destra di cui abbiamo parlato sulle pagine precedenti.
La linea 50 richiede quante volte si desidera richiamare la routine di scroll.
La 60 stampa una stringa nella nona riga e memoriza il 9 nella locazione 50000.
La 70 definisce l'indirizzo di partenza della routine in linguaggio macchina.

```
10 CLEAR200, 39999! : RESTORE : IN=40000!
20 READ A$
30 N=VAL("&H" + A$) : IFN>255THEN GOTO50 ELSE POKEIN,N
40 IN=IN + 1 : GOTO20
50 INPUT "QUANTE POSIZIONI" ;NP : SCREEN1 : KEYOFF
60 LOCATE0,9,0 : PRINT "PROVIAMO LA ROUTINE DI SCROLL" : POKE 50000!,9
70 DEFUSR=40000!
80 FORI=1TONP : A=USR(0) : FORP=1TO77 : NEXT : NEXT
90 GOTO 50
100 DATA 3A, 50, C3, 01, 20, 00, 21, 1F, 18, 3C, 3D, CA, 52, 9C, 09, 4A, 9C, CD, 4A, 00,
    F5
110 DATA 01, 1F, 00, 2B, CD, 4A, 00, 23, CD, 4D, 00, 2B, 0B, 79, B0, C2, 59, 9C, F1, CD,
    4D, 00, C9,, FFF
```

La 80 racchiude due cicli: il primo, quello delle chiamate ognuna delle quali sposterà a destra di una posizione tutta la riga nove, l'altro per infrappore un ritardo affinché la scritta sia leggibile (il linguaggio macchina è troppo veloce!).
Alla fine del ciclo un GOTO porta nuovamente alla riga 50.
Per uscire è necessario un CTRL/STOP.

Ovviamente la stringa è sostituibile.
Un programma del genere, magari con qualche perfezionamento, può essere indicato per far girare dei messaggi importanti o delle scritte pubblicitarie.
C'è solo un inconveniente: non siamo abituati a leggere da destra a sinistra!
Per rimediare, ecco il disassemblato della routine di scroll a sinistra.
Questa è stata posizionata a partire da 40100 in modo che sia possibile eventualmente utilizzarla insieme a quella di scorrimento a destra (in 40000).
Nota che il numero di riga su cui operare deve essere posto nella cella 50001.

SCROLL DI UNA RIGA A SINISTRA

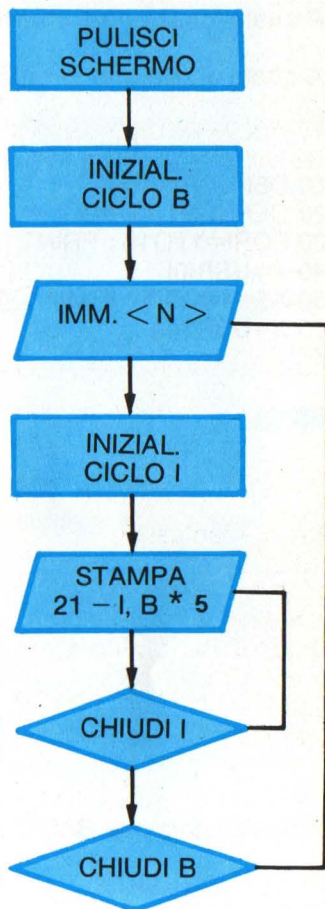
Con questo programma puoi mettere alla prova la routine di scroll a sinistra.
La riga interessata e la undicesima.

```
10 CLEAR200, 39999! : IN=40100!
20 READ A$
30 N=VAL("&H" +A$) : IFN>255THEN 50 ELSE POKE IN,N
40IN=IN + 1:GOTO 20
50 INPUT "QUANTE POSIZIONI" ;NP : SCREEN1 : KEYOFF
60 LOCATE0,11,0 : PRINT "QUESTA VOLTA MUOVE A SINISTRA" : POKE 50001!,
   11
70 DEFUSR1=40100!
80 FORI=1TONP : A=USR1(0) : FORP=1TO77 : NEXT : NEXT
90 GOTO 50
100 DATA 3A, 50, C3, 01, 20, 00, 21, 1F, 18, 3C, 3D, CA, 52, 9C, 09, C3, 4A, 9C, CD, 4A,
   00, F5
110 DATA 01, 1F, 00, 2B, CD, 4A, 00, 23, CD, 4D, 00, 2B, 0B, 79, B0, C2, 59, 9C, F1, CD,
   4D, 00, C9, FFF
```

Alla tua fantasia e iniziativa, invece, un programma che utilizzi, magari premendo un tasto, ora la routine di scroll a destra, ora quello a sinistra.
Non è difficile: provaci!

INDIRIZZO	CODICE HEX		ASSEMBLY	
			ORG	40100
9CA4	3A51C3		LD	A,(50001)
9CA7	012000		LD	BC,32
9CAA	210018		LD	HL,6144
9CAD	3C		INC	A
9CAE	3D	LOOP:	DEC	A
9CAF	CAB69C		JP	Z,SAVEP
9CB2	09		ADD	HL,BC
9CB3	C3AE9C		JP	LOOP
9CB6	CD4A00	SAVEP:	CALL	74
9CB9	F5		PUSH	AF
9CBA	011F00		LD	BC,31
9CBD	23	LOOP1:	INC	HL
9CBE	CD4A00		CALL	74
9CC1	2B		DEC	HL
9CC2	CD4D00		CALL	77
9CC5	23		INC	HL
9CC6	0B		DEC	BC
9CC7	79		LD	A,C
9CC8	B0		OR	B
9CC9	C2BD9C		JP	NZ,LOOP1
9CCC	F1		POP	AF
9CCD	CD4D00		CALL	77
9CD0	C9		RET	

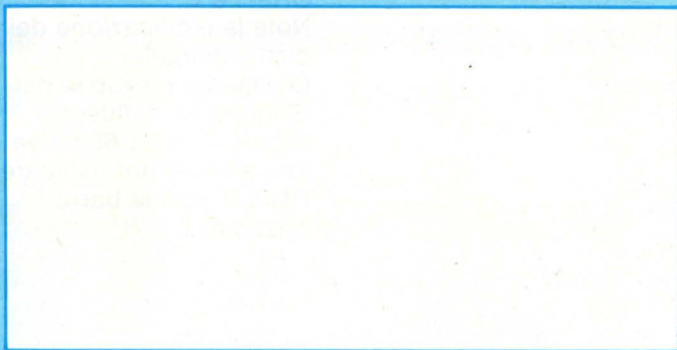
Rappresentazione grafica

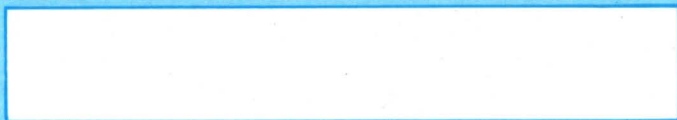
Ecco un programma per generare un diagramma a barre. Lo schermo visualizza fino a 6 elementi verticali in grado di rappresentare valori compresi tra 0 e 34. Il carattere utilizzato per stampare le barre è ottenuto premendo GRPH e P.
Nota la nidificazione dei cicli automatici facilmente rilevabile dal diagramma di flusso.
Il ciclo in linea 65 serve unicamente per allineare l'INPUT con le barre orizzontali.

```
10 CLS : SCREEN0 : WIDTH40
20 FORB=0TO5
30 INPUT"N=";N
40 FORI=1TON
50 LOCATE40 - I,B*3,0 : PRINT "  "
60 NEXTI
65 FORV=1TO2 : PRINT : NEXT V
70 NEXTB
```

VIDEOESERCIZI

A quale locazione è chiamata la routine che disabilita lo schermo?

A quale quella che lo riabilita?

```
10 DEFUSR=&H41
20 DEFUSR1=&H44
30 FORI=1TO16 : PRINT "vediamo un po' se si vede"; I : NEXT
40 X=USR(0)
50 A$=INKEY$ : IFA$=" "THEN50
60 X=USR1(0)
```

A quale istruzione BASIC è analoga la routine chiamata da questi programmi?

```
10 DEFUSR=&HCC
20 X=USR(0)
```

```
10 DEFUSR1=&HCF
20 X=USR1(0)
```

```
10 DEFUSR2=&HC0
20 X=USR2(0)
```